



# Learning Iterative Processes with Recurrent Neural Networks to Correct Satellite Image Classification Maps

Emmanuel Maggiori, Guillaume Charpiat, Yuliya Tarabalka, Pierre Alliez

## ► To cite this version:

Emmanuel Maggiori, Guillaume Charpiat, Yuliya Tarabalka, Pierre Alliez. Learning Iterative Processes with Recurrent Neural Networks to Correct Satellite Image Classification Maps. 2016. hal-01388551

**HAL Id: hal-01388551**

**<https://inria.hal.science/hal-01388551>**

Preprint submitted on 27 Oct 2016

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Learning Iterative Processes with Recurrent Neural Networks to Correct Satellite Image Classification Maps

Emmanuel Maggiori  
Inria - TITANE

emmanuel.maggiori@inria.fr

Guillaume Charpiat  
Inria - TAO

Yuliya Tarabalka  
Inria - TITANE

Pierre Alliez  
Inria - TITANE

## Abstract

*While initially devised for image categorization, convolutional neural networks (CNNs) are being increasingly used for the pixelwise semantic labeling of images. However, the proper nature of the most common CNN architectures makes them good at recognizing but poor at localizing objects precisely. This problem is magnified in the context of aerial and satellite image labeling, where a spatially fine object outlining is of paramount importance.*

*Different iterative enhancement algorithms have been presented in the literature to progressively improve the coarse CNN outputs, seeking to sharpen object boundaries around real image edges. However, one must carefully design, choose and tune such algorithms. Instead, our goal is to directly learn the iterative process itself. For this, we formulate a generic iterative enhancement process inspired from partial differential equations, and observe that it can be expressed as a recurrent neural network (RNN). Consequently, we train such a network from manually labeled data for our enhancement task. In a series of experiments we show that our RNN effectively learns an iterative process that significantly improves the quality of satellite image classification maps.*

## 1. Introduction

One of the most explored problems in remote sensing is the pixelwise labeling of satellite imagery. Such a labeling is used in a wide range of practical applications, such as precision agriculture and urban planning. Recent technological developments have substantially increased the availability and resolution of satellite data. Besides the computational complexity issues that arise, these advances are posing new challenges in the processing of the images. Notably, the fact that large surfaces are covered introduces a significant variability in the appearance of the objects. In addition, the fine details in high resolution images make it difficult to classify the pixels from elementary cues. For example, the different

parts of an object often contrast more with each other than with other objects [1]. Using high-level contextual features thus plays a crucial role at distinguishing object classes.

Convolutional neural networks (CNNs) [15] are receiving an increasing attention, due to their ability to automatically discover relevant contextual features in image categorization problems. CNNs have already been used in the context of remote sensing [22, 27], featuring powerful recognition capabilities. However, when the goal is to label images at the pixel level, the output classification maps are too coarse. For example, buildings are successfully detected but their boundaries in the classification map rarely coincide with the real object boundaries. We can identify two main reasons for this coarseness in the classification:

a) There is a structural limitation of CNNs to carry out fine-grained classification. If we wish to keep a low number of learnable parameters, the ability to learn long-range contextual features comes at the cost of losing spatial accuracy, i.e., a trade-off between detection and localization. This is a well-known issue and still a scientific challenge [5, 19].

b) In the specific context of remote sensing imagery, there is a significant lack of spatially accurate reference data for training. For example, the OpenStreetMap collaborative database provides large amounts of free-access maps over the earth, but irregular misregistrations and omissions are frequent all over the dataset. In such circumstances, CNNs cannot do better than learning rough estimates of the objects' locations, given that the boundaries are hardly located on real edges in the training set.

Let us remark that in the particular context of high-resolution satellite imagery, the spatial precision of the classification maps is of paramount importance. Objects are small and a boundary misplaced by a few pixels significantly hampers the overall classification quality. In other application domains, such as semantic segmentation of natural scenes, while there have been recent efforts to better shape the output objects, a high resolution output seems to be less of a priority. For example, in the popular Pascal VOC semantic segmentation dataset, there is a band of several unlabeled pixels around the objects, where accuracy is

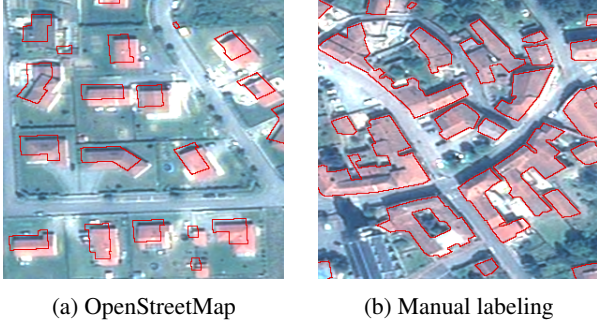


Figure 1: Samples of reference data for the *building* class. Imprecise OpenStreetMap data vs manually labeled data.

not computed to assess the performance of the methods.

There are two recent tendencies to overcome the structural issues that lead to coarse classification maps. One of them is to use new types of CNN architectures, specifically designed for pixel labeling, that seek to address the detection/localization trade-off. For example, Noh et al. [23] duplicate a base classification CNN by attaching a reflected “deconvolution” network, which learns to upsample the coarse classification maps. Another tendency is to use first the base CNN as a rough classifier of the objects’ locations, and then process this classification using the original image as guidance, so that the output objects better align to real image edges. For example, Zheng et al. [31] use a fully connected CRF in this manner, and Chen et al. [4] diffuse the classification probabilities with an edge-stopping function based on image features.

The first scheme seems unfeasible in the context of large-scale satellite imagery, due to the lack of large amounts of precisely labeled training data. Even if an advanced architecture could eventually learn to conduct a more precise labeling, this is not useful when the training data itself is inaccurate. We thus here adopt the second strategy, reinjecting image information to an enhancement module that sharpens the coarse classification maps around the objects. To train or set the parameters of this enhancement module, as well as to validate the algorithms, we assume we can afford to manually label small amounts of data. In Fig. 1a we show an example of imprecise data to which we have access in large quantities, and in Fig. 1b we show a portion of manually labeled data. In our approach, the first type of data is used to train a large CNN to learn the generalities of the object classes, and the second to tune and validate the algorithm that enhances the coarse classification maps outputted by the CNN.

An algorithm to enhance coarse classification maps would require, on the one hand, to define the image features to which the objects must be attached. This is data-dependent, not every image edge being necessarily an object boundary. On the other hand, we must also decide which enhancement algorithm to use, and tune it. Besides

the efforts that this requires, we could also imagine that the optimal approach would go beyond the algorithms presented in the literature. For example we could perform different types of corrections on the different classes, based on the type of errors that are often present in each of them.

Our goal is to create a system that learns the appropriate enhancement algorithm itself, instead of designing it by hand. This involves learning not only the relevant features but also the rationale behind the enhancement technique, thus intensively leveraging the power of machine learning.

To achieve this, we first formulate a generic partial differential equation governing a broad family of iterative enhancement algorithms. This generic equation conveys the idea of progressively refining a classification map based on local cues, yet it does not provide the specifics of the algorithm. We then observe that such an equation can be expressed as a combination of common neural network layers, whose learnable parameters define the specific behavior of the algorithm. We then see the whole iterative enhancement process as a recurrent neural network (RNN).

The RNN is provided with a small piece of manually labeled image, and trained end to end to improve coarse classification maps. It automatically discovers relevant data-dependent features to enhance the classification as well as the equations that govern every enhancement iteration.

## 1.1. Related work

A common way to tackle the aerial image labeling problem is to use classifiers such as support vector machines [2] or neural networks [21] on the individual pixel spectral signatures (i.e., a pixel’s “color” but not limited to RGB bands). In some cases, a few neighboring pixels are analyzed jointly to enhance the prediction and enforce the spatial smoothness of the output classification maps [9]. Hand-designed features such as textural features have also been used [18]. The use of an iterative classification enhancement process on top of hand-designed features has also been explored in the context of image labeling [26].

Following the recent advent of deep learning and to address the new challenges posed by large-scale aerial imagery, Penatti et al. [24] used CNNs to assign aerial image patches to categories (e.g., ‘residential’, ‘harbor’) and Vakalopoulou et al. [27] addressed building detection using CNNs. Mnih [22] and Maggiori et al. [20] used CNNs to learn long-range contextual features to produce classification maps. These networks require some degree of down-sampling in order to consider large contexts with a reduced number of parameters. They perform well at detecting the presence of objects but do not outline them accurately.

Our work can also be related to the area of natural image semantic segmentation. Notably, fully convolutional networks (FCN) [19] are becoming increasingly popular to conduct pixelwise image labeling. FCN networks are made

up of a stack of convolutional and pooling layers followed by so-called deconvolutional layers that upsample the resolution of the classification maps, possibly combining features at different scales. The output classification maps being too coarse, the authors of the Deeplab network [5] added a fully connected conditional random field (CRF) on top of both the FCN and the input color image, in order to enhance the classification maps.

Zheng et al. [31] recently reformulated the fully connected CRF of Deeplab as an RNN, and Chen et al. [4] designed an RNN that emulates the domain transform filter [10]. Such a filter is used to sharpen the classification maps around image edges, which are themselves detected with a CNN. In these methods the refinement algorithm is designed beforehand and only few parameters that rule the algorithm are learned as part of the network’s parameters. The innovating aspect of these approaches is that both steps (coarse classification and enhancement) can be seen as a single end-to-end network and optimized simultaneously.

Instead of predefining the algorithmic details as in previous works, we formulate a general iterative refinement algorithm through an RNN and let the network learn the specific algorithm. To our knowledge, little work has explored the idea of learning an iterative algorithm. In the context of image restoration, the preliminary work by Liu et al. [16, 17] proposed to optimize the coefficients of a linear combination of predefined terms. Chen et al. [6] later modeled this problem as a diffusion process and used an RNN to learn the linear filters involved as well as the coefficients of a parametrized nonlinear function. Our problem is however different, in that we use the image as guidance to update a classification map, and not to restore the image itself. Besides, while we drew inspiration on diffusion processes, we are also interested in imitating other iterative processes like active contours, thus we do not restrict our system to diffusions but consider all PDEs.

## 2. Enhancing classification maps with RNNs

Let us assume we are given a set of score (or “heat”) maps  $u_k$ , one for each possible class  $k \in \mathcal{L}$ , in a pixelwise labeling problem. The score of a pixel reflects the likelihood of belonging to a class, according to the classifier’s predictions. The final class assigned to every pixel is the one with maximal value  $u_k$ . Alternatively, a softmax function can be used to interpret the results as probability scores:  $P(k) = e^{u_k} / \sum_{j \in \mathcal{L}} e^{u_j}$ . Fig. 2 shows a sample of the type of fuzzy heat map outputted by a CNN in the context of satellite image classification, for the class ‘building’.

Our goal is to combine the score maps  $u_k$  with information derived from the input image (e.g., edge features) to sharpen the scores near the real objects in order to enhance the classification.

One way to perform such a task is to progressively en-

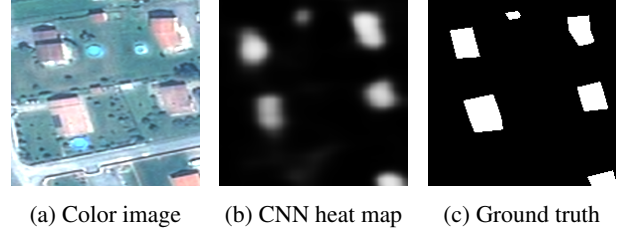


Figure 2: Sample classification of buildings with a CNN.

hance the score maps by using partial differential equations (PDEs). In this section we first describe different types of PDEs we could certainly imagine to design in order to solve our problem. Instead of discussing which one is the best, we then propose a generic iterative process to enhance the classification maps without specific constraints on the algorithm rationale. Finally, we show how this equation can be expressed and trained as a recurrent neural network (RNN).

### 2.1. Partial differential equations (PDEs)

We can formulate a variety of diffusion processes applied to the maps  $u_k$  as partial differential equations. For example, the heat flow is described as:

$$\frac{\partial u_k(x)}{\partial t} = \text{div}(\nabla u_k(x)), \quad (1)$$

where  $\text{div}(\cdot)$  denotes the divergence operator in the spatial domain of  $x$ . Applying such a diffusion process in our context would smooth out the heat maps. Instead, our goal is to design an image-dependent smoothing process that aligns the heat maps to the image features. A natural way of doing this is to modulate the gradient in Eq. 1 by a scalar function  $g(x, I)$  that depends on the input image  $I$ :

$$\frac{\partial u_k(x)}{\partial t} = \text{div}(g(I, x) \nabla u_k(x)). \quad (2)$$

Eq. 2 is similar to the Perona-Malik diffusion [25] with the exception that Perona-Malik uses the smoothed function itself to guide the diffusion.  $g(I, x)$  denotes an edge-stopping function that takes low values near borders of  $I(x)$  in order to slow down the smoothing process there.

Another possibility would be to consider a more general variant in which  $g(I, x)$  is replaced by a matrix  $D(I, x)$ , acting as a diffusion tensor that redirects the flow based on image properties instead of just slowing it down near edges:

$$\frac{\partial u_k(x)}{\partial t} = \text{div}(D(I, x) \nabla u_k(x)). \quad (3)$$

This formulation relates to the so-called anisotropic diffusion process [29].

Alternatively, one can draw inspiration from the level set framework. For example, the geodesic active contours tech-

nique formulated as level sets translates into:

$$\frac{\partial u_k(x)}{\partial t} = |\nabla u_k(x)| \operatorname{div} \left( g(I, x) \frac{\nabla u_k(x)}{|\nabla u_k(x)|} \right). \quad (4)$$

Such a formulation favors the zero level set to align with minima of  $g(I, x)$  [3]. Schemes based on Eq. 4 could then be used to improve heat maps  $u_k$ , provided they are scaled so that segmentation boundaries match zero levels.

As shown above, many different PDE approaches can be devised to enhance classification maps. However, several choices must be made to select the appropriate PDE and tailor it to our problem.

For example, one must choose the edge-stopping function  $g(I, x)$  in Eqs. 2, 4. Common choices are exponential or rational functions on the image gradient [25], which in turn requires to set an edge-sensitivity parameter. Extensions to the original Perona-Malik approach could also be considered, such as a popular regularized variant that computes the gradient on a Gaussian-smoothed version of the input image [29]. In the case of opting for anisotropic diffusion, one must design  $D(I, x)$ .

Instead of using trial and error to perform such design, our goal is to let a machine learning approach discover by itself a useful iterative process for our task.

## 2.2. A generic classification enhancement process

PDEs are usually discretized in space by using finite differences, which represent derivatives as discrete convolution filters. We build upon this scheme to write a generic discrete formulation of an enhancement iterative process.

Let us consider that we take as input a score map  $u_k$  (for class  $k$ ) and, in the most general case, an arbitrary number of feature maps  $\{g_1, \dots, g_p\}$  derived from image  $I$ . In order to perform differential operations, of the type  $\{\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial^2}{\partial x \partial y}, \frac{\partial^2}{\partial x^2}, \dots\}$ , we consider convolution kernels  $\{M_1, M_2, \dots\}$  and  $\{N_1^j, N_2^j, \dots\}$  to be applied to the heat map  $u_k$  and to the features  $g_j$  derived from image  $I$ , respectively. While we could certainly directly provide a bank of filters  $M_i$  and  $N_i^j$  in the form of Sobel operators, Laplacian operators, etc., we may simply let the system learn the required filters. We group all the feature maps that result from applying these convolutions, in a single set:

$$\Phi(u_k, I) = \{M_i * u_k, N_l^j * g_j(I); \forall i, j, l\}. \quad (5)$$

Let us now define a generic discretized scheme as:

$$\frac{\partial u_k(x)}{\partial t} = f_k(\Phi(u_k, I)(x)), \quad (6)$$

where  $f_k$  is a function that takes as input the values of all the features in  $\Phi(u_k, I)$  at an image point  $x$ , and combines them. While convolutions  $M_i$  and  $N_i^j$  convey the “spatial”

reasoning, e.g., gradients,  $f_k$  captures the combination of these elements, such as the products in Eqs. 2 and 4.

Instead of deriving an arbitrary number of possibly complex features  $N_i^j * g_j(I)$  from image  $I$ , we can think of a simplified scheme in which we directly operate on  $I$ , by considering only convolutions:  $N_i * I$ . The list of functions considered in Eq. 6 is then

$$\Phi(u_k, I) = \{M_i * u_k, N_j * I; \forall i, j\} \quad (7)$$

and consists only of convolutional kernels directly applied to the heat maps  $u_k$  and to the image  $I$ . From now on, we here stick to this simpler formulation, yet we acknowledge that it might be eventually useful to work on a higher-level representation rather than on the input image itself. Note that if one restricts functions  $f_k$  in Eq. 6 to be linear, we still obtain the set of all linear PDEs. We consider *any* function  $f_k$ , introducing non-linearities.

PDEs are usually discretized in time, taking the form:

$$u_{k,t+1}(x) = u_{k,t}(x) + \delta u_{k,t}(x), \quad (8)$$

where  $\delta u_{k,t}$  denotes the overall update of  $u_{k,t}$  at time  $t$ .

Note that the convolution filters in Eqs. 5 and 7 are class-agnostic:  $M_i$ ,  $N_j$  and  $N_l^j$  do not depend on  $k$ , while  $f_k$  may be a different function for each class  $k$ . Function  $f_k$  thus determines the contribution of each feature to the equation, contemplating the case in which a different evolution might be optimal for each of the classes, even if just in terms of a time-step factor. In the next section we detail a way to learn the update functions  $\delta u_{k,t}$  from training data.

## 2.3. Iterative processes as RNNs

We now show that the generic iterative process can be implemented as an RNN, and thus trained from labeled data. This stage requires to provide the system with a piece of accurately labeled ground truth (see e.g., Fig. 1b).

Let us first show that one iteration, as defined in Eqs. 6-8, can be expressed in terms of common neural network layers. Let us focus on a single pixel for a specific class, simplifying the notation from  $u_{k,t}(x)$  to  $u_t$ . Fig. 3 illustrates the proposed network architecture. Each iteration takes as input the image  $I$  and a given heat map  $u_t$  to enhance at time  $t$ . In the first iteration,  $u_t$  is the initial coarse heat map to be improved, outputted by another pre-trained neural network in our case. From the heat map  $u_t$  we derive a series of filter responses, which correspond to  $M_i * u_t$  in Eq. 7. These responses are found by computing the dot product between a set of filters  $M_i$  and the values of  $u_{k,t}(\cdot)$  in a spatial neighborhood of a given point. Analogously, a set of filter responses are computed at the same spatial location on the input image, corresponding to the different  $N_j * I$  of Eq. 7. These operations are convolutions when performed

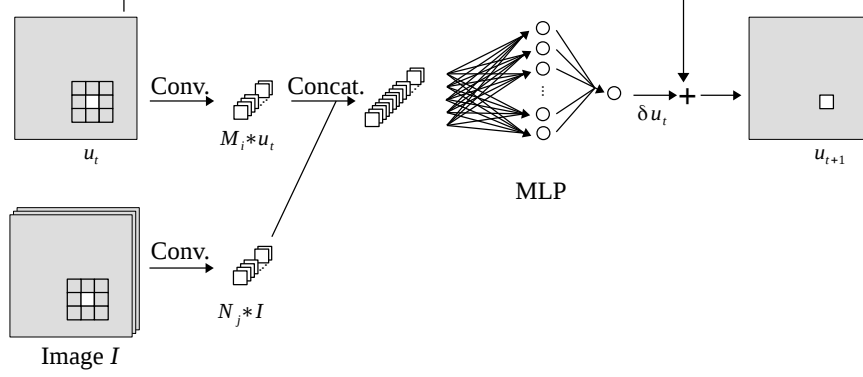


Figure 3: One enhancement iteration represented as common neural network layers.

densely in space,  $N_j * I$  and  $M_i * u_t$  being feature maps of the filter responses.

These filters are then “concatenated”, forming a pool of features  $\Phi$  coming from both the input image and the heat map, as in Eq. 7, and inputted to  $f_k$  in Eq. 6. We must now learn the function  $\delta u_t$  that describes how the heat map  $u_t$  is updated at iteration  $t$  (cf. Eq. 8), based on these features.

Eq. 6 does not introduce specifics about function  $f_k$ . In (1)-(4), for example, it includes products between different terms, but we could certainly imagine other functions. We therefore model  $\delta u_t$  through a multilayer perceptron (MLP), because it can approximate any function within a bounded error. We include one hidden layer with nonlinear activation functions followed by an output neuron with a linear activation (a typical configuration for regression problems), although other MLP architectures could be used. Applying this MLP densely is equivalent to performing convolutions with  $1 \times 1$  kernels at every layer. The implementation to densely label entire images is then straightforward.

The value of  $\delta u_t$  is then added to  $u_t$  in order to generate the updated map  $u_{t+1}$ . This addition is performed pixel by pixel in the case of a dense input. Note that although we could have removed this addition and let the MLP directly output the updated map  $u_{t+1}$ , we opted for this architecture since it is more closely related to the equations and better conveys the intention of a progressive refinement of the classification map. Moreover, learning  $\delta u_t$  instead of  $u_{t+1}$  has a significant advantage at training time: a random initialization of the networks’ parameters centered around zero means that the initial RNN represents an iterative process close to the identity (with some noise). Training uses the asymmetry induced by this noise to progressively move from the identity to a more useful iterative process.

The overall iterative process is implemented by unrolling a finite number of iterations, as illustrated in Fig. 4, under the constraint that the parameters are shared among all iterations. Such sharing is enforced at training time by a simple modification to the back-propagation training algorithm where the derivatives of every instance of a weight at dif-

ferent iterations are averaged [30]. Note that issues with vanishing or exploding gradients may arise when too many iterations are unrolled, an issue inherent to deep network architectures. Note also that the spatial features are shared across the classes, while a different MLP is learned for each of them, following Eq. 6. As depicted by Fig. 4 and conveyed in the equations, the features extracted from the input image are independent of the iteration.

The RNN of Fig. 4 represents then a dynamical system that iteratively improves the class heat maps. Training such an RNN amounts to finding the optimal dynamical system for our enhancement task.

### 3. Implementation details

We first describe the CNN used to produce the coarse predictions, then detail our RNN. The network architecture was implemented using Caffe deep learning library [13].

Our coarse prediction CNN is based on a previous remote sensing network presented by Mnih [22]. We create a fully convolutional [19] version of Mnih’s network, since recent remote sensing work has shown the theoretical and practical advantages of this type of architecture [14, 20]. The CNN takes 3-band color image patches at  $1\text{m}^2$  resolution and produces as many heat maps as classes considered. The resulting four-layer architecture is as follows: 64 conv. filters ( $12 \times 12$ , stride 4)  $\rightarrow$  128 conv. filters ( $3 \times 3$ )  $\rightarrow$  128 conv. filters ( $3 \times 3$ )  $\rightarrow$  3 conv. filters ( $9 \times 9$ ). Since the first convolution is performed with a stride of 4, the resulting feature maps have a quarter of the input resolution. Therefore, a deconvolutional layer [19] is added on top to upsample the classification maps to the original resolution. The activation functions used in the hidden layers are rectified linear units. This network is trained on patches randomly selected from the training dataset. We group 64 patches with classification maps of size  $64 \times 64$  into minibatches (following [22]) to estimate the gradient of the network’s parameters and back-propagate them. Our loss function is the cross-entropy between the target and predicted class probabilities. Stochastic gradient descent is used for

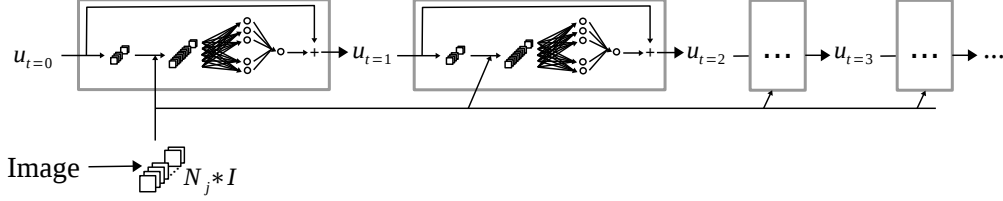


Figure 4: Modules of Fig. 3 are stacked (while sharing parameters) to implement an RNN.

optimization, with learning rate 0.01, momentum 0.9 and an L2 weight regularization of 0.0002. We did not however optimized these parameters nor the networks’ architectures.

We now detail the implementation of the RNN described in Sec. 2.3. We unroll five RNN iterations and learn 32  $M_i$  and 32  $N_j$  filters, both of spatial dimensions  $5 \times 5$ . As explained in Sec. 2.3, an independent MLP is learned for every class, using 32 hidden neurons each and with rectified linear activations. Training is performed on random patches and with the cross-entropy loss function, as done with the coarse CNN. The employed gradient descent algorithm is AdaGrad [8], which exhibits a faster convergence in our case, using a base learning rate of 0.01 (higher values make the loss diverge). All weights are initialized randomly by sampling from a distribution that depends on the number of neuron inputs [11]. We trained the RNN for 50,000 iterations, until observing convergence of the training loss, which took around four hours on a single GPU.

## 4. Experiments

We perform our experiments on images acquired by a Pléiades satellite over the area of Forez, France. An RGB color image is used, obtained by pansharpening [28] the satellite data, which provides a spatial resolution of  $0.5 \text{ m}^2$ . Since the networks described in Sec. 3 are specifically designed for images with a  $1 \text{ m}^2$  resolution, we downsample the Pléiades images before feeding them to our networks and bilinearly upsample the outputs.

From this image we selected an area with OpenStreetMap (OSM) [12] coverage to create a  $22.5 \text{ km}^2$  training dataset for the classes *building*, *road* and *background*. The reference data was obtained by rasterizing the raw OSM maps. Misregistrations and omissions are present all over the dataset (see Fig. 1a and further examples in suppl. material). Buildings tend to be misaligned or omitted, while many roads in the ground truth are not visible in the image (or the other way around). This is the dataset used to train the initial coarse CNNs.

We manually labeled two  $2.25 \text{ km}^2$  tiles to train and test the RNN at enhancing the predictions of the coarse network. We denote them by enhancement and test sets, respectively. Note that our RNN system must discover an algorithm to refine an existing classification map, and not to conduct the classification itself, hence a smaller training set should be

sufficient for this stage.

In the following, we report the results obtained by using the proposed method on the Pléiades dataset. Fig. 5 provides closeups of results on different fragments of the test dataset. The initial and final maps (before and after the RNN enhancement) are depicted, as well as the intermediate results through the RNN iterations. We show both a set of final classification maps and some single-class fuzzy probability maps. We can observe that as the RNN iterations go by, the classification maps are refined and the objects better align to image edges. The fuzzy probabilities become more confident, sharpening object boundaries. To quantitatively assess this improvement we compute two measures on the test set: the overall accuracy (proportion of correctly classified pixels) and the intersection over union (IoU) [19]. Mean IoU has become the standard in semantic segmentation since it is more reliable in the presence of imbalanced classes (such as *background* class, which is included to compute the mean) [7]. As summarized in the table of Fig. 6(a), the performance of the original coarse CNN (denoted by CNN) is significantly improved by attaching our RNN (CNN+RNN). Both measures increase monotonously along the intermediate RNN iterations, as depicted in Fig. 6(b).

The initial classification of roads has an overlap of less than 10% with the roads in the ground truth, as shown by its individual IoU. The RNN makes them emerge from the background class, now overlapping the ground truth roads by over 50%. Buildings also become better aligned to the real boundaries, going from less than 40% to over 70% overlap with the ground truth buildings. This constitutes a multiplication of the IoU by a factor of 5 for roads and 2 for buildings, which indicates a significant improvement at outlining and not just detecting objects.

Additional visual fragments before and after the RNN refinement are shown in Fig. 7. We can observe in the last row how the iterative process learned by the RNN both thickens and narrows the roads depending on the location.

We also compare our RNN to the approach in [5] (here denoted by CNN+CRF), where a fully-connected CRF is coupled both to the input image and the coarse CNN output, in order to refine the predictions. This is the idea behind the so-called Deeplab network, which constitutes one of the most important current baselines in the semantic seg-



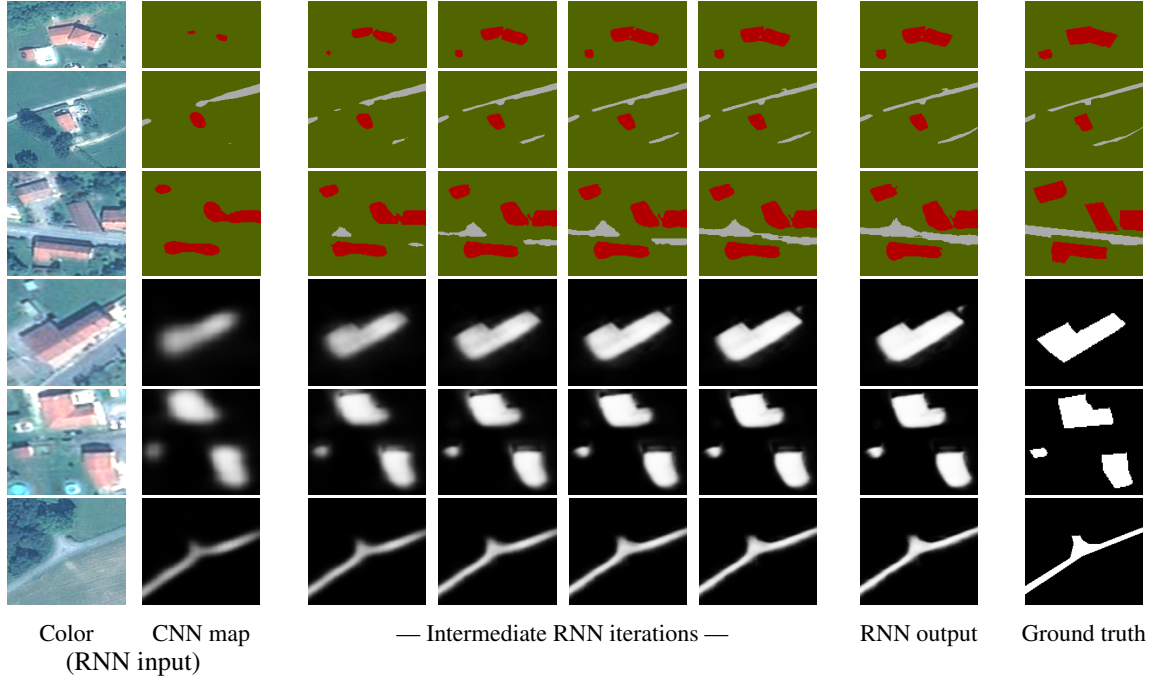
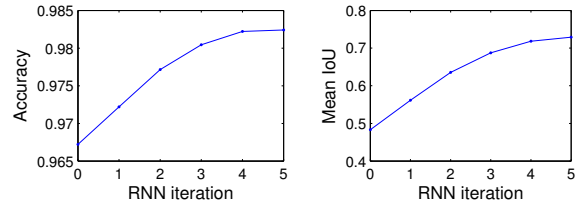


Figure 5: Evolution of fragments of classification maps (top rows) and single-class fuzzy scores (bottom rows) through RNN iterations.

Method	Overall accuracy	Mean IoU	Class-specific IoU		
			Build.	Road	Backg.
CNN	96.72	48.32	38.92	9.34	96.69
CNN+CRF	96.96	44.15	29.05	6.62	96.78
CNN+RNN <sup>≡</sup>	97.78	65.30	59.12	39.03	97.74
CNN+RNN	<b>98.24</b>	<b>72.90</b>	<b>69.16</b>	<b>51.32</b>	<b>98.20</b>

(a) Numerical comparison (in %)



(b) Evolution through RNN iterations

Figure 6: Quantitative evaluation on Pléiades images test set over Forez, France.

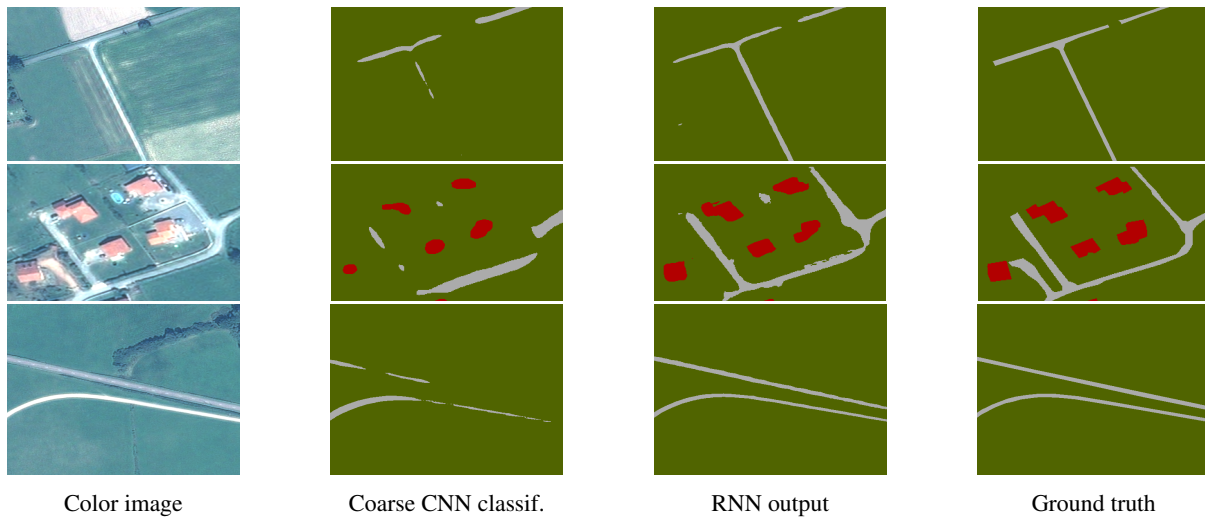


Figure 7: Initial coarse classifications and the enhanced maps by using RNNs.



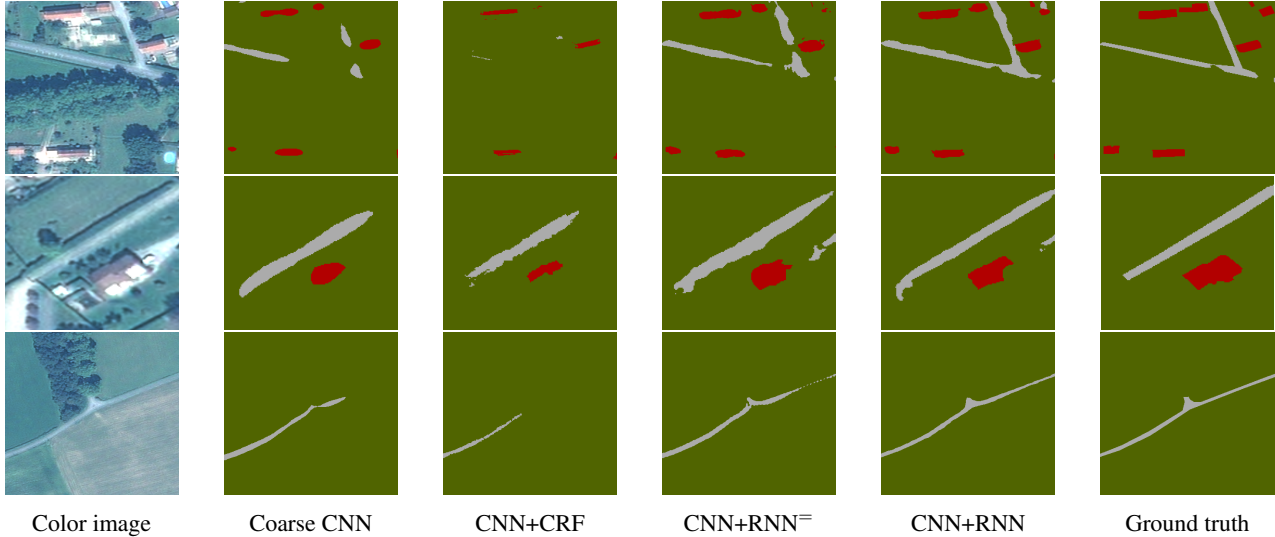


Figure 8: Visual comparison on closeups of the Pléiades dataset.

mentation community. While the CRF itself could also be implemented as an RNN [31], we here stick to the original formulation because the CRF as RNN idea is only interesting if we want to train the system end to end (i.e., together with the coarse prediction network). In our case we wish to leave the coarse network as is, otherwise we risk overfitting it to this much smaller set. We thus simply use the CRF as in [5] and tune the energy parameters by performing a grid search using the enhancement set as a reference. Five iterations of inference on the fully-connected CRF were performed in every case.

To further analyze our method, we also consider an alternative enhancement RNN in which the weights of the MLP are shared across the different classes (which we denote by  $\text{CNN+RNN}^=$ ). This forces the system to learn the same function to update all the classes, instead of a class-specific function.

Numerical results are included in the table of Fig. 6(a) and visual fragments are compared in Fig. 8. The  $\text{CNN+CRF}$  approach does sharpen the maps but this often occurs around the wrong edges. It also makes small objects disappear in favor of larger objects (usually the background class) when edges are not well marked, which explains the mild increase in overall accuracy but the decrease in mean IoU. While the  $\text{CNN+RNN}^=$  outperforms the CRF, both quantitative and visual results are beaten by the  $\text{CNN+RNN}$ , supporting the importance of learning a class-specific enhancement function. The suppl. material includes the visualization of different filters learned by the RNN and visual results over a larger surface.

To validate the importance of using a recurrent architecture, and following Zheng et al. [31], we retrained our system considering every iteration of the RNN as an independent step with its own parameters. After training for the

same number of iterations, it yields a lower performance on the test set compared to the RNN and a higher performance on the training set. If we keep on training, the non-recurrent network still enhances its training accuracy while performing poorly on the test set, implying a significant degree of overfitting with this variant of the architecture. This provides evidence that constraining our network to learn an iterative enhancement process is crucial for its success.

## 5. Concluding remarks

In this work we presented an RNN that learns how to refine the coarse output of another neural network, in the context of pixelwise image labeling. The inputs are both the coarse classification maps to be corrected and the original color image. The output at every RNN iteration is an update to the classification map of the previous iteration, using the color image for guidance.

Little human intervention is required, since the specifics of the refinement algorithm are not provided by the user but learned by the network itself. For this, we analyzed different iterative alternatives and devised a general formulation that can be interpreted as a stack of common neuron layers. At training time, the RNN discovers the relevant features to be taken both from the classification map and from the input image, as well as the function that combines them.

The experiments on satellite imagery show that the classification maps are improved significantly, increasing the overlap of the foreground classes with the ground truth, and outperforming other approaches by a large margin. Thus, the proposed method not only detects but also outlines the objects. To conclude, we demonstrated that RNNs succeed in learning iterative processes for classification enhancement tasks.

## Acknowledgment

All Pléiades images are ©CNES (2012 and 2013), distribution Airbus DS / SpotImage. The authors would like to thank the CNES for initializing and funding the study, and providing Pléiades data.

## References

- [1] H. G. Akçay and S. Aksoy. Building detection using directional spatial constraints. In *IEEE IGARSS*, pages 1932–1935, 2010.
- [2] G. Camps-Valls and L. Bruzzone. Kernel-based methods for hyperspectral image classification. *IEEE TGRS*, 43(6):1351–1362, 2005.
- [3] V. Caselles, R. Kimmel, and G. Sapiro. Geodesic active contours. *IJCV*, 22(1):61–79, 1997.
- [4] L.-C. Chen, J. T. Barron, G. Papandreou, K. Murphy, and A. Yuille. Semantic image segmentation with task-specific edge detection using cnns and a discriminatively trained domain transform. *arXiv preprint arXiv:1511.03328*, 2015.
- [5] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *ICLR*, May 2015.
- [6] Y. Chen, W. Yu, and T. Pock. On learning optimized reaction diffusion processes for effective image restoration. In *IEEE CVPR*, pages 5261–5269, 2015.
- [7] G. Csurka, D. Larlus, F. Perronnin, and F. Meylan. What is a good evaluation measure for semantic segmentation?. In *BMVC*, 2013.
- [8] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12:2121–2159, 2011.
- [9] M. Fauvel, Y. Tarabalka, J. A. Benediktsson, J. Chanussot, and J. C. Tilton. Advances in spectral-spatial classification of hyperspectral images. *Proceedings of the IEEE*, 101(3):652–675, 2013.
- [10] E. S. L. Gastal and M. M. Oliveira. Domain transform for edge-aware image and video processing. *ACM Trans. Graph.*, 30(4):69:1–69:12, 2011.
- [11] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International conference on artificial intelligence and statistics*, pages 249–256, 2010.
- [12] M. Haklay and P. Weber. Openstreetmap: User-generated street maps. *Pervasive Computing, IEEE*, 7(4):12–18, 2008.
- [13] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.
- [14] M. Kampffmeyer, A.-B. Salberg, and R. Jenssen. Semantic segmentation of small objects and modeling of uncertainty in urban remote sensing images using deep convolutional neural networks. In *IEEE CVPR Workshops*, pages 1–9, 2016.
- [15] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [16] R. Liu, Z. Lin, W. Zhang, and Z. Su. Learning PDEs for image restoration via optimal control. In *ECCV*, pages 115–128. Springer, 2010.
- [17] R. Liu, Z. Lin, W. Zhang, K. Tang, and Z. Su. Toward designing intelligent pdes for computer vision: An optimal control approach. *Image and vision computing*, 31(1):43–56, 2013.
- [18] C. Lloyd, S. Berberoglu, P. J. Curran, and P. Atkinson. A comparison of texture measures for the per-field classification of mediterranean land cover. *International Journal of Remote Sensing*, 25(19):3943–3965, 2004.
- [19] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *IEEE CVPR*, 2015.
- [20] E. Maggiori, Y. Tarabalka, G. Charpiat, and P. Alliez. Fully convolutional neural networks for remote sensing image classification. In *IEEE IGARSS*, 2016.
- [21] J. Mas and J. Flores. The application of artificial neural networks to the analysis of remotely sensed data. *International Journal of Remote Sensing*, 29(3):617–663, 2008.
- [22] V. Mnih. *Machine learning for aerial image labeling*. PhD thesis, University of Toronto, 2013.
- [23] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *IEEE ICCV*, pages 1520–1528, 2015.
- [24] O. Penatti, K. Nogueira, and J. Santos. Do deep features generalize from everyday objects to remote sensing and aerial scenes domains? In *IEEE CVPR Workshops*, pages 44–51, 2015.
- [25] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE TPAMI*, 12(7):629–639, 1990.
- [26] Z. Tu. Auto-context and its application to high-level vision tasks. In *IEEE CVPR*, pages 1–8. IEEE, 2008.
- [27] M. Vakalopoulou, K. Karantza, N. Komodakis, and N. Paragios. Building detection in very high resolution multispectral data with deep learning features. In *IEEE IGARSS*, pages 1873–1876, 2015.
- [28] Z. Wang, D. Ziou, C. Armenakis, D. Li, and Q. Li. A comparative analysis of image fusion methods. *IEEE TGRS*, 43(6):1391–1402, 2005.
- [29] J. Weickert. *Anisotropic diffusion in image processing*. Teubner Stuttgart, 1998.
- [30] P. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.
- [31] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. Torr. Conditional random fields as recurrent neural networks. In *IEEE CVPR*, pages 1529–1537, 2015.